# Spatial Exploration

*Christian Stratton*

*May 22, 2018*

## Spatial Exploration

### Simple Gaussian Process

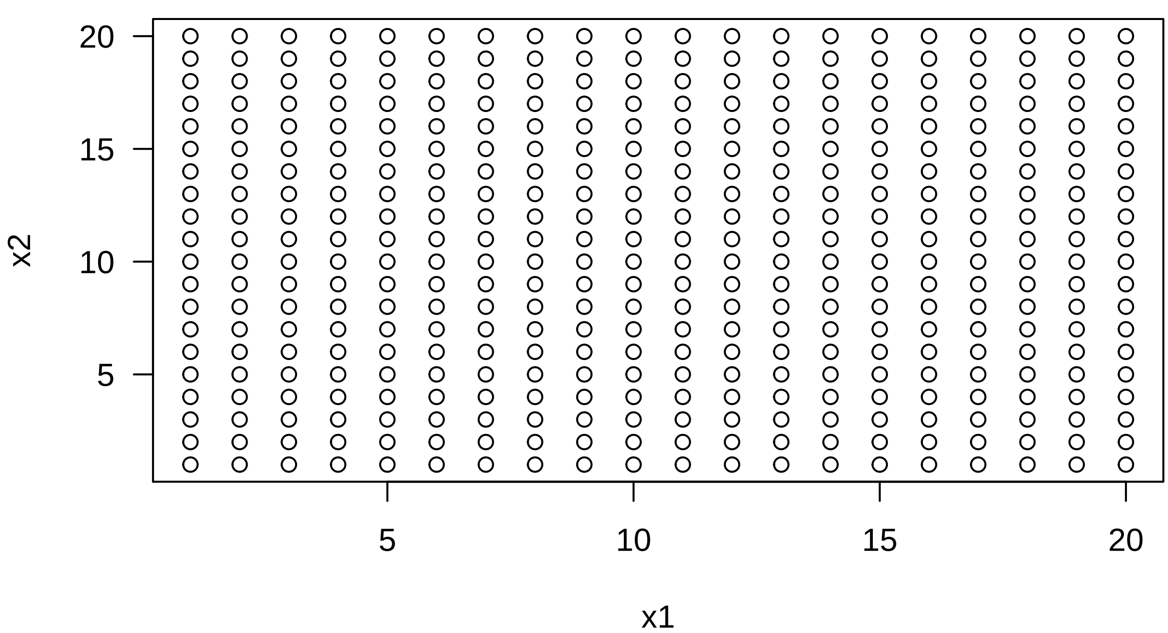
Consider the model:

$$\boldsymbol{y}|\boldsymbol{\theta} \sim MVN(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2\boldsymbol{H}(\phi))$$

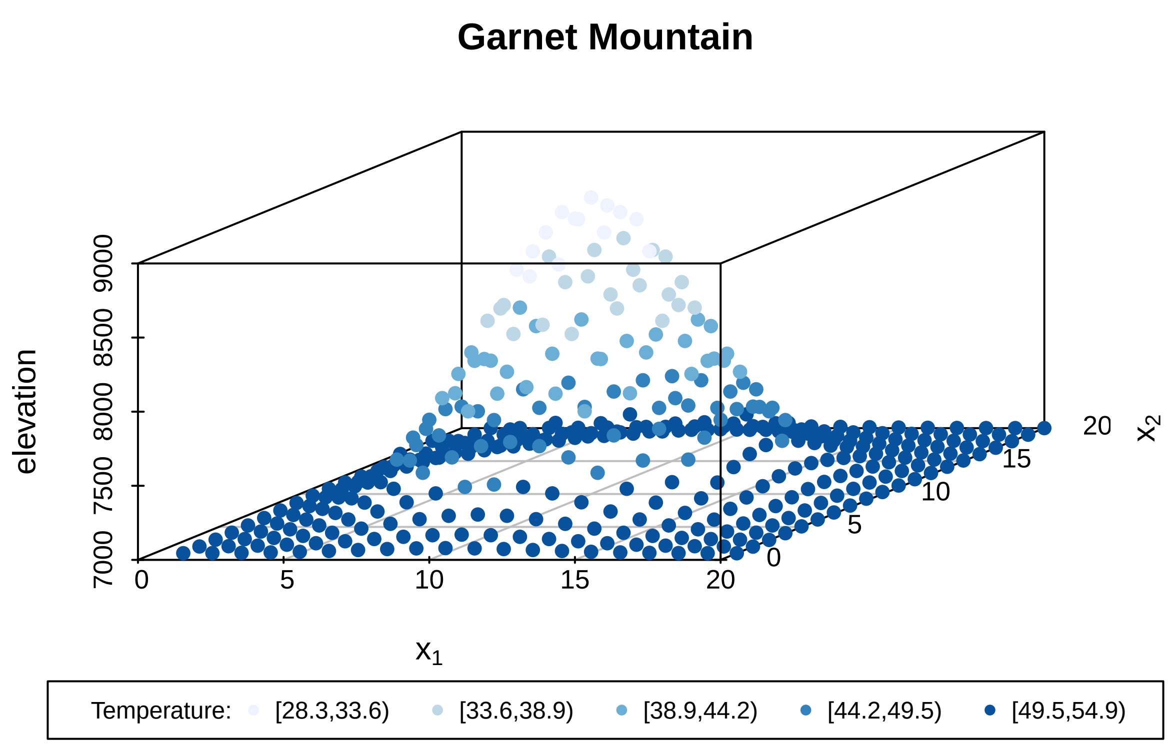
where $H(\phi)$ is the simple exponential case. That is, $\boldsymbol{H}(\phi) = exp\left(-\frac{d_{ij}}{\phi}\right)$.

#### Generate Spatially Correlated Data

```r
set.seed(298032)
grid <- data.frame(expand.grid(1:20, 1:20))
names(grid) <- c("x1", "x2")
plot(grid, las = 1)
```

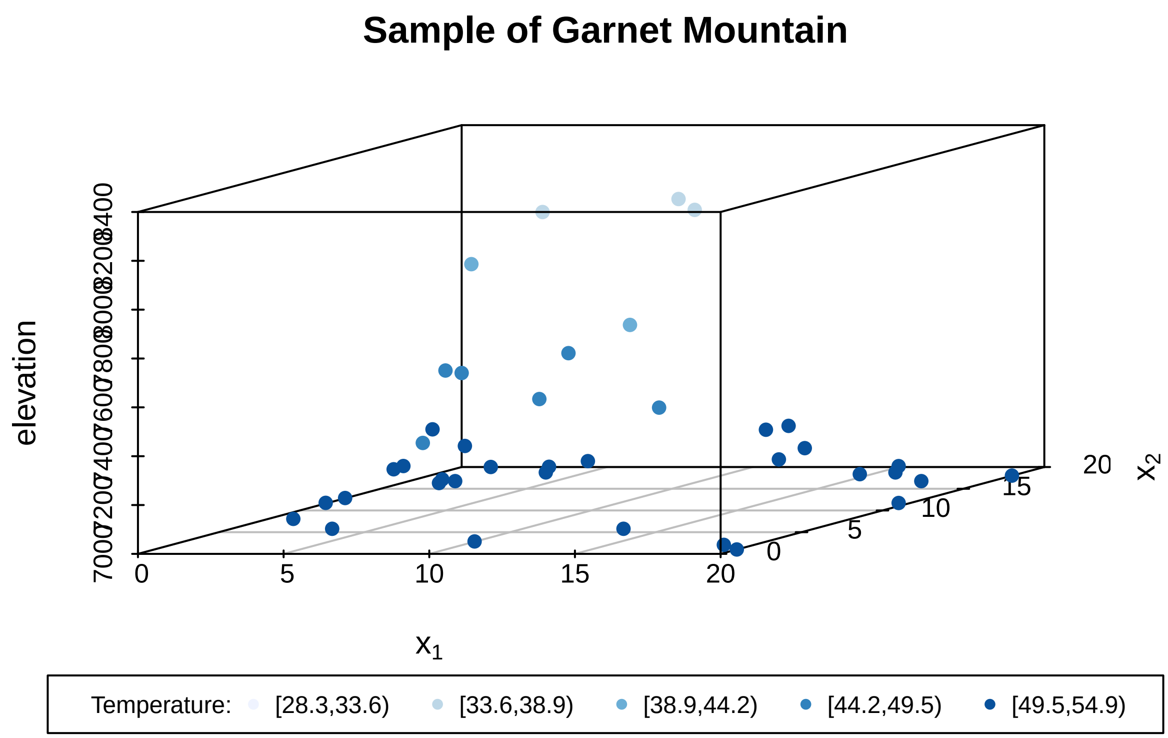


```r
sp.grid <- grid
coordinates(sp.grid) <- ~x1 + x2
d <- rgeos::gDistance(sp.grid, byid = T)

# create spatial covariate - elevation
grid$elevation <- with(grid, 2000 * exp(-(1/20) * ((x1 - 10)^2 + (x2 - 10)^2)) +
    7000)

# create response - temperature #b0 = 140, b1 = -1/80, sigma2 = 1, phi = 10
temp <- with(grid, rmvnorm(1, mean = 140 - (1/80) * elevation, sigma = 1 * exp(-d/10)))

# final spatial df
garnet.df <- data.frame(cbind(grid, temp = as.numeric(temp)))
MyPalette <- brewer.pal(5, "Blues")
palette(MyPalette)
split.obs <- cut(garnet.df$temp, 5, right = F)
cuts <- levels(split.obs)

par(mar = c(10, 4, 4, 2) + 0.1)
scatterplot3d::scatterplot3d(x = grid$x1, y = grid$x2, z = grid$elevation, pch = 16,
    color = MyPalette[as.numeric(split.obs)], main = "Garnet Mountain", xlab = expression("x"[1]),
    ylab = expression("x"[2]), zlab = "elevation", bg = "red")
legend("bottom", legend = c("Temperature:", cuts), pch = c(NA, rep(16, 5)),
    col = c("white", MyPalette), inset = -0.35, xpd = T, horiz = T, cex = 0.75)
```




```r
# check it out plot3d(x = garnet.df$x1, y = garnet.df$x2, z =
# garnet.df$elevation, main = 'Garnet Mountain', col =
```

```
# MyPalette[as.numeric(split.obs)], size = 5, xlab = '', ylab = '', zlab =
# '') rgl.bbox(xlen = 0, ylen = 0, color = c('orange')) legend3d('topright',
# legend = cuts, pch = 16, col = MyPalette, cex=.7, inset=c(0.05))
# snapshot3d(filename = '3dplot.png', fmt = 'png')

# sample points
set.seed(16240)
id <- sample(400, 40)

samp <- garnet.df[sort(id), ]
split.samp.obs <- cut(samp$temp, c(28.3, 33.6, 38.9, 44.2, 49.5, 54.9), right = F)

scatterplot3d::scatterplot3d(x = samp$x1, y = samp$x2, z = samp$elevation, pch = 16,
    color = MyPalette[as.numeric(split.samp.obs)], main = "Sample of Garnet Mountain",
    xlab = expression("x"[1]), ylab = expression("x"[2]), zlab = "elevation",
    bg = "red")
legend("bottom", legend = c("Temperature:", cuts), pch = c(NA, rep(16, 5)),
    col = c("white", MyPalette), inset = -0.35, xpd = T, horiz = T, cex = 0.75)
```




**Sampler**

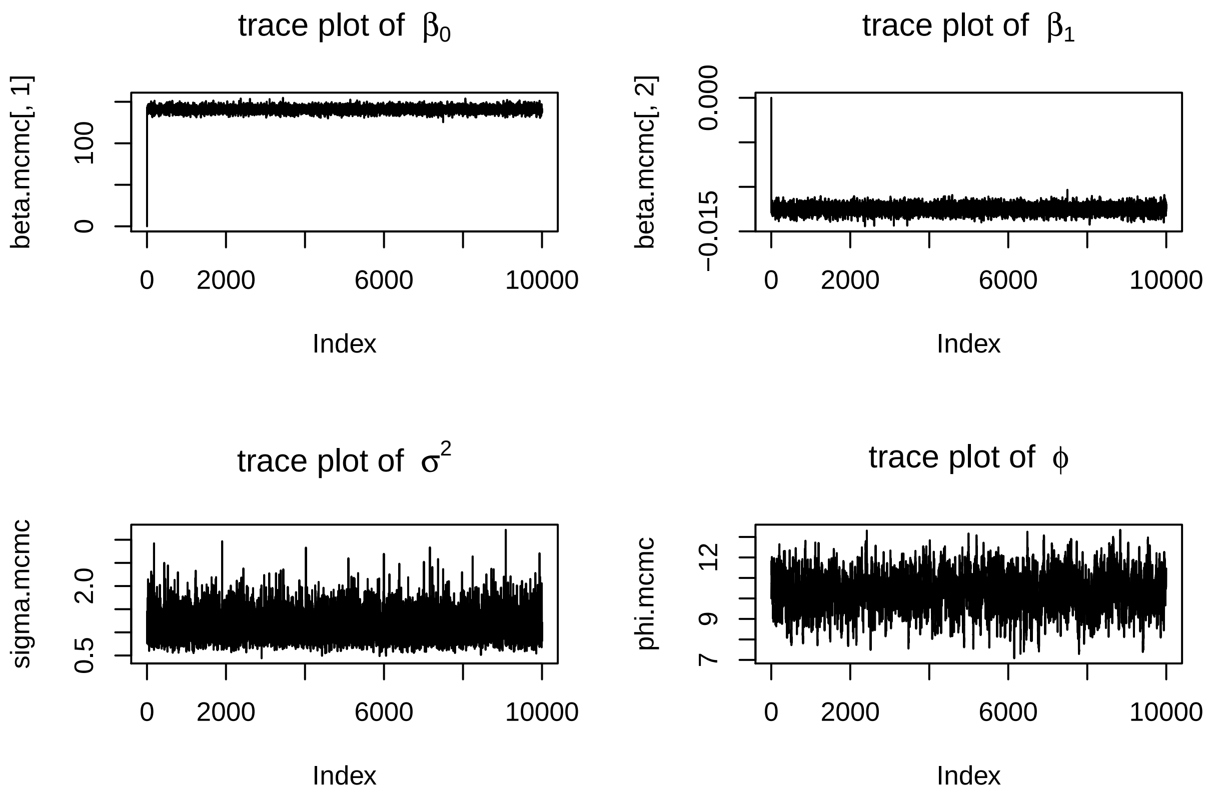
Priors:

$$p(\boldsymbol{\beta}) = \text{mvnormal}\left(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right)$$
$$p(\sigma^2) = \text{inverse-gamma}(\frac{\nu_0}{2}, \frac{\sigma_0^2}{2})$$
$$p(\phi) = \text{gamma}(a, b)$$

Note: Ended up placing a normal$(10, 1)$ prior on $\phi$... very informative, couldn't get it to behave otherwise.

```
# priors
n <- 40
mu0 <- rep(0, 2)
Sigma0 <- 1000 * diag(2)
nu0 <- 1e-04
sigma20 <- 10
# nu0 <- 6 sigma20 <- 4
a <- 1e-04
b <- 1e-04
Sigma0.inv <- solve(Sigma0)

sp.samp.grid <- data.frame(x1 = samp$x1, x2 = samp$x2)
coordinates(sp.samp.grid) <- ~x1 + x2
dist.mat <- rgeos::gDistance(sp.samp.grid, byid = T)

y <- samp$temp
X <- model.matrix(~elevation, data = samp)

# setup sampler
num.mcmc <- 10000
step.size <- 0.5

beta.mcmc <- matrix(0, num.mcmc, 2)
sigma.mcmc <- rep(0, num.mcmc)
phi.mcmc <- rep(0, num.mcmc)
accept.ratio <- rep(0, num.mcmc)

# initialize sampler
sigma.mcmc[1] <- 1
phi.mcmc[1] <- 10

for (i in 2:num.mcmc) {
    # gibbs step for beta and sigma

    # sample beta | sigma, phi
    Sigma <- sigma.mcmc[i - 1] * exp(-dist.mat/phi.mcmc[i - 1])
    Sigma.inv <- solve(Sigma)

    A <- solve(t(X) %*% Sigma.inv %*% X + Sigma0.inv)
    B <- t(X) %*% Sigma.inv %*% y - Sigma0.inv %*% mu0

    beta.mcmc[i, ] <- mvtnorm::rmvnorm(1, mean = A %*% B, sigma = A)

    # sample sigma | beta, phi
    resid <- y - X %*% beta.mcmc[i, ]
    H <- exp(-dist.mat/phi.mcmc[i - 1])
```

```r
    H.inv <- solve(H)

    SSR <- t(resid) %*% H.inv %*% resid

    sigma.mcmc[i] <- LearnBayes::rigamma(1, (nu0 + n)/2, (SSR + sigma20)/2)

    # metropolis step for phi
    phi.s <- phi.mcmc[i - 1]
    phi.star <- phi.mcmc[i - 1] + rnorm(1, mean = 0, sd = sqrt(step.size))

    num <- mvtnorm::dmvnorm(y, mean = X %*% beta.mcmc[i, ], sigma = sigma.mcmc[i] *
        exp(-dist.mat/phi.star), log = T) + dnorm(phi.star, mean = 10, sd = 1,
        log = T)
    # dunif(phi.star, 0, max(dist.mat), log = T) #dgamma(phi.star, a, 1 / b, log
    # = T) #dnorm(phi.star, mean = 10, sd = 1, log = T) #dexp(phi.star, 1/a, log
    # = T)
    denom <- mvtnorm::dmvnorm(y, mean = X %*% beta.mcmc[i, ], sigma = sigma.mcmc[i] *
        exp(-dist.mat/phi.s), log = T) + dnorm(phi.s, mean = 10, sd = 1, log = T)
    # dunif(phi.s, 0, max(dist.mat), log = T) #dgamma(phi.s, a, 1 / b, log = T)
    # #dnorm(phi.s, mean = 10, sd = 1, log = T) #dexp(phi.s, 1/a, log = T)

    log.r <- num - denom

    if (log(runif(1)) < log.r) {
        phi.mcmc[i] <- phi.star
        accept.ratio[i] <- 1
    } else {
        phi.mcmc[i] <- phi.s
    }

    if (i%%500 == 0) {
        message("Progress: ", bquote(.(i)), "th iteration of chain")
        message(bquote(.(round(i/(num.mcmc), 5) * 100)), "% through chain")
    }
}

par(mfrow = c(2, 2))
plot(beta.mcmc[, 1], type = "l", main = expression("trace plot of " ~ beta[0]))
plot(beta.mcmc[, 2], type = "l", main = expression("trace plot of " ~ beta[1]))
plot(sigma.mcmc, type = "l", main = expression("trace plot of " ~ sigma^2))
plot(phi.mcmc, type = "l", main = expression("trace plot of " ~ phi))
```




```
mean(accept.ratio)
```

```
## [1] 0.7658
```

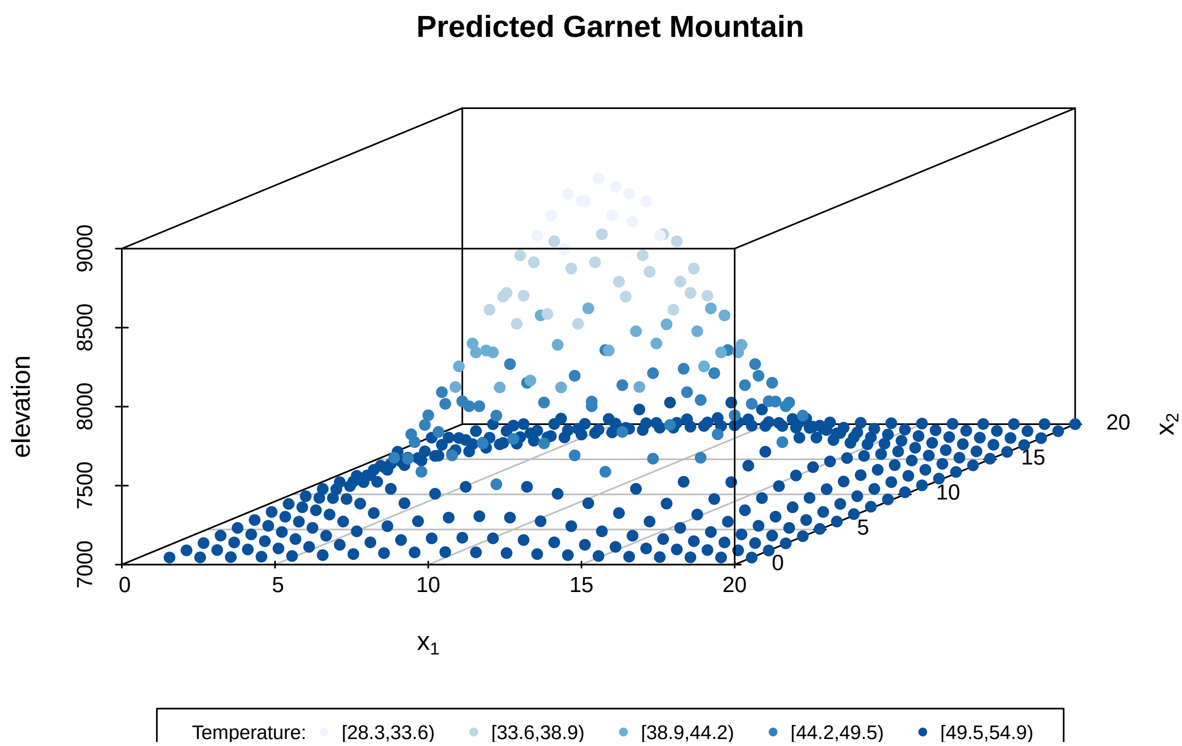
**Posterior Predictive Checks**

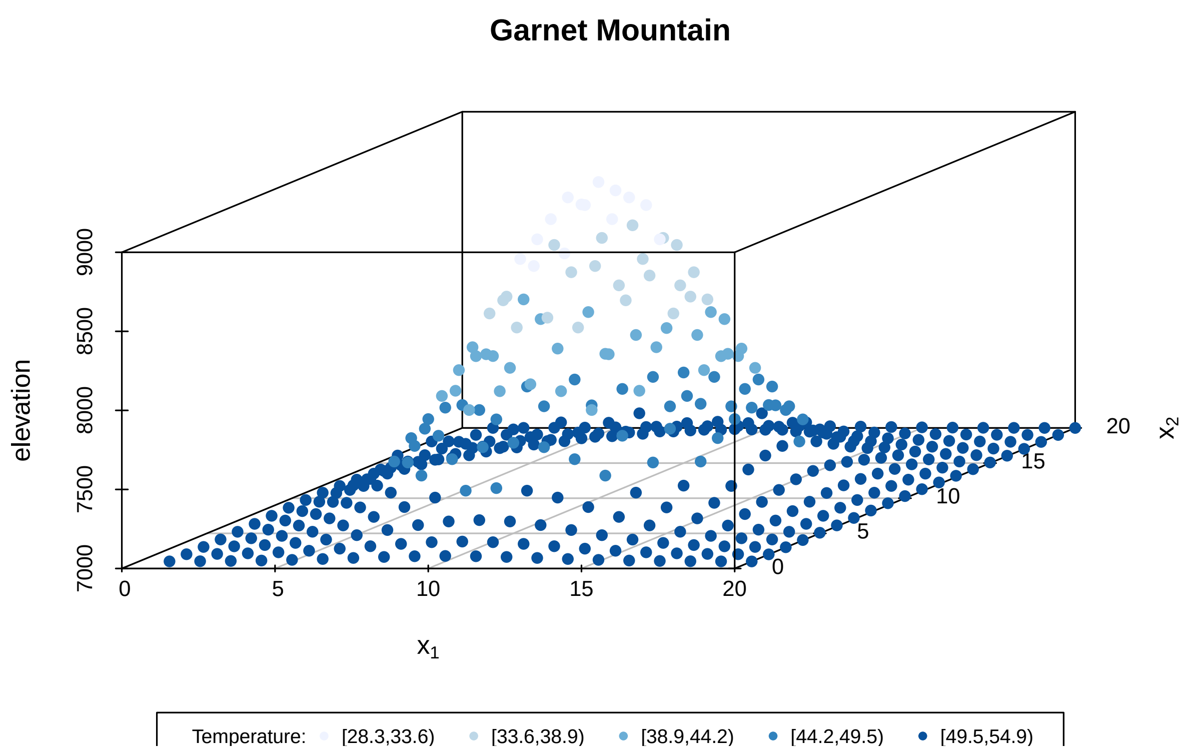
To predict across a vector of unobserved locations, $S_0 = \{s_{01}, s_{02}, ..., s_{0m}\}$, we have:

$$p(\boldsymbol{y}_0|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}_0) = \int p(\boldsymbol{y}_0|\boldsymbol{y}, \boldsymbol{\theta}, \boldsymbol{X}_0)p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X})d\boldsymbol{\theta} \approx \frac{1}{G}\sum_{g=1}^{G} p(\boldsymbol{y}_0|\boldsymbol{y}, \boldsymbol{\theta}^{(g)}, \boldsymbol{X}_0)$$

```
# predict across population grid
X0 <- model.matrix(~elevation, data = garnet.df)

# posterior predictive distribution
pred <- matrix(0, num.mcmc, nrow(garnet.df))
for (i in 1:dim(beta.mcmc)[1]) {
    mean <- X0 %*% beta.mcmc[i, ]
    H <- exp(-d/phi.mcmc[i])
    var <- sigma.mcmc[i] * H

    pred[i, ] <- rmvnorm(1, mean = mean, sigma = var)

    if (i%%500 == 0) {
        message("Progress: ", bquote(.(i)), "th iteration")
```

```
        message(bquote(.(round(i/(num.mcmc), 5) * 100)), "% through")
    }
}

pred.burn <- pred[500:dim(pred)[1], ]
pred.temp <- colMeans(pred.burn)

pred.df <- cbind(garnet.df[, 1:3], pred.temp)
split.pred.obs <- cut(pred.df$pred.temp, c(28.3, 33.6, 38.9, 44.2, 49.5, 54.9),
    right = F)

par(mfrow = c(2, 1))

scatterplot3d::scatterplot3d(x = pred.df$x1, y = pred.df$x2, z = pred.df$elevation,
    pch = 16, color = MyPalette[as.numeric(split.pred.obs)], main = "Predicted Garnet Mountain",
    xlab = expression("x"[1]), ylab = expression("x"[2]), zlab = "elevation",
    bg = "red")
legend("bottom", legend = c("Temperature:", cuts), pch = c(NA, rep(16, 5)),
    col = c("white", MyPalette), inset = -0.35, xpd = T, horiz = T, cex = 0.75)

scatterplot3d::scatterplot3d(x = grid$x1, y = grid$x2, z = grid$elevation, pch = 16,
    color = MyPalette[as.numeric(split.obs)], main = "Garnet Mountain", xlab = expression("x"[1]),
    ylab = expression("x"[2]), zlab = "elevation", bg = "red")
legend("bottom", legend = c("Temperature:", cuts), pch = c(NA, rep(16, 5)),
    col = c("white", MyPalette), inset = -0.35, xpd = T, horiz = T, cex = 0.75)
```




```
SSR <- sum((garnet.df$temp - pred.df$pred.temp)^2)
```